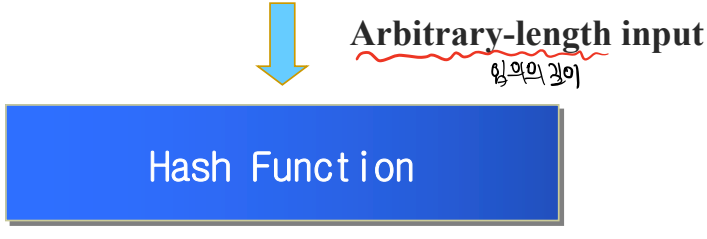
Cryptographic Hash Function

Jong Hwan Park

Cryptographic Hash Function (1)

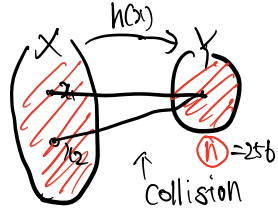
- Hash function
  - In data structures : O(1) insertion and lookup times (storing h(x) || x)
  - In cryptography



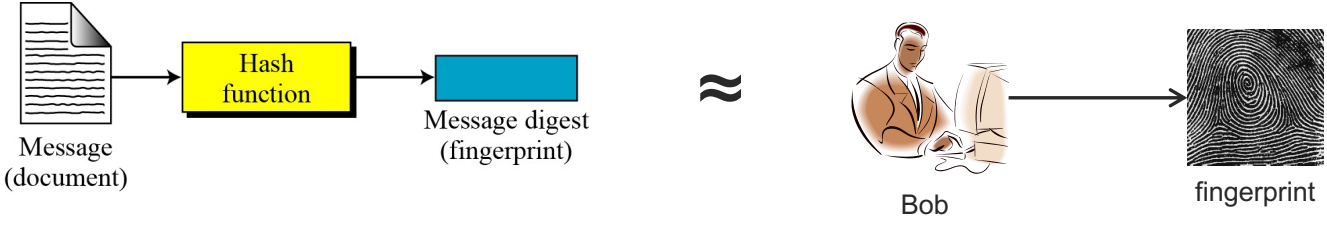
Hash output = Message digest (fingerprint)

Cryptographic Hash Function (2)

- In CHF, collision resistance is a mandatory requirement
  - A collision in CHF is a pair of inputs (x, x') such that h(x) = h(x')
  - Function $h: \{0,1\}^{*} \rightarrow \{0,1\}^{n}$ is collision-resistant if it is *infeasible* to find a collision in h
    - Collisions must exist (why?)
    - Polynomial time (feasible) vs. exponential time (infeasible)

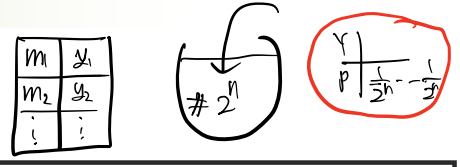


- Using CHF(with collision resistance)
  - (*long* message M, h(M)) ≈ (person, fingerprint)
  - h(M) is the electronic equivalence of fingerprint



Cryptographic Hash Function (3)

- Uniformly distributed



1 bit change

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

01	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

Hash Function

Hash Function

49 16 D6 BD B7 F7 8E 68 03 69
8C AB 32 D1 58 6E A4 57 DF C8

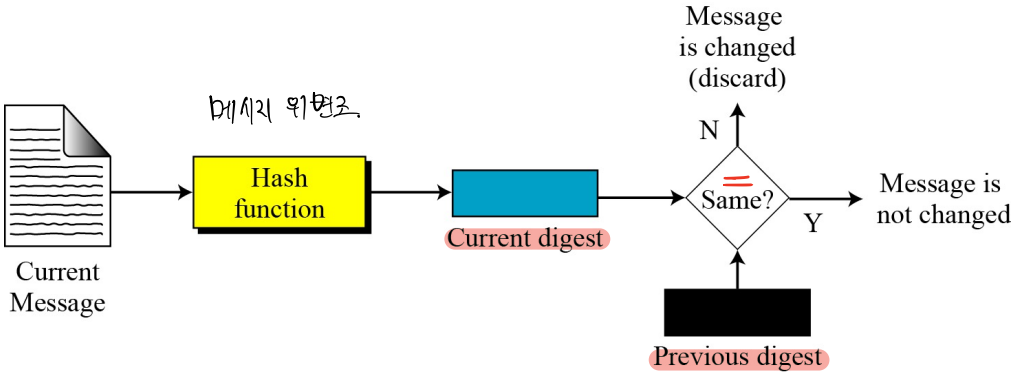
≠

52 FB EC 10 72 00 59 86 D1 A7
EF B6 5B 04 71 41 A1 14 7A FF

Cryptographic Hash Function (4)

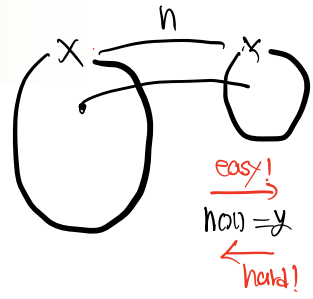
- Using CHF (with CR and also uniform distribution)
  - Can detect whether a file is changed or not
  - A file-storage system, gathering of e-evidence, etc

block chain 에서 메시지 위변조 check.

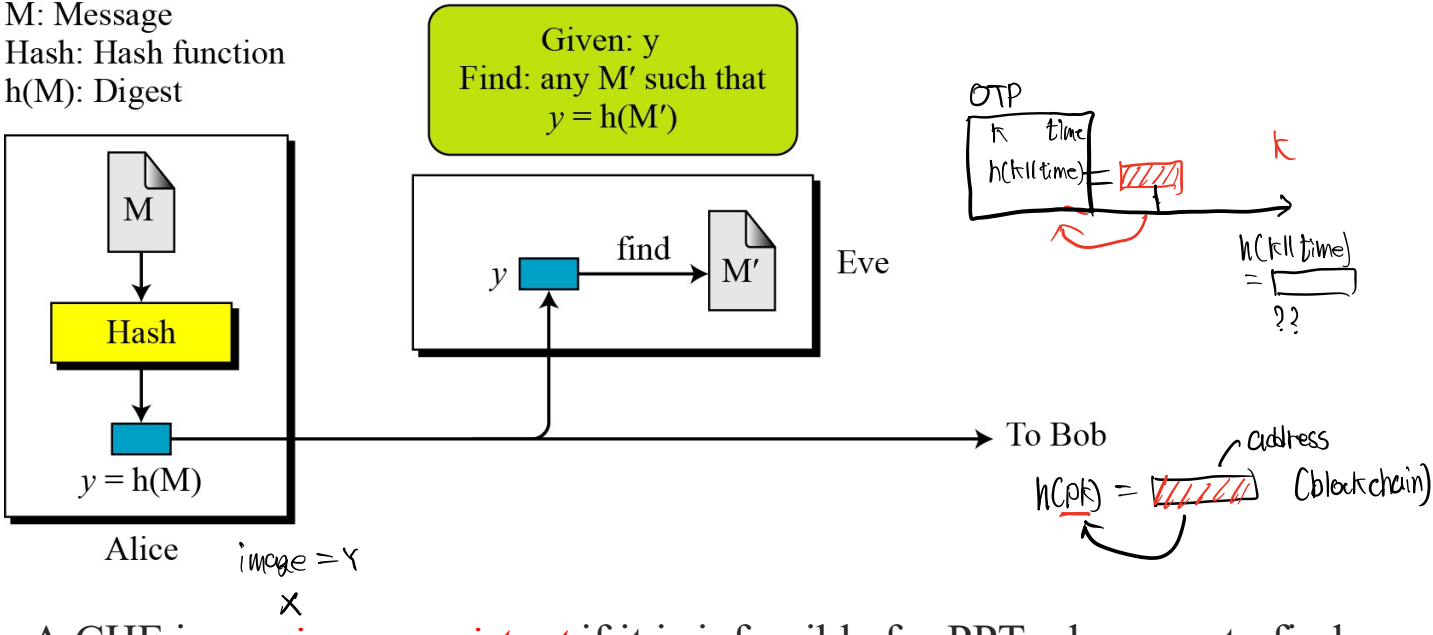


- Security requirements for CHF
  - (uniformly distributed) 균일하게 분포.
  - Pre-image resistance (one way)
  - Second pre-image resistance
  - Collision resistance

Requirements for CHF (1)

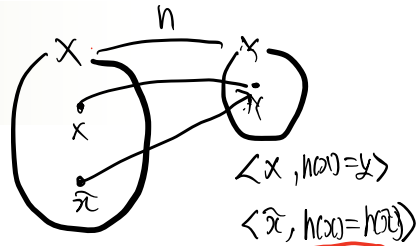


- Pre-image resistance (one wayness)
  - In real scenario (e.g., one-time password)

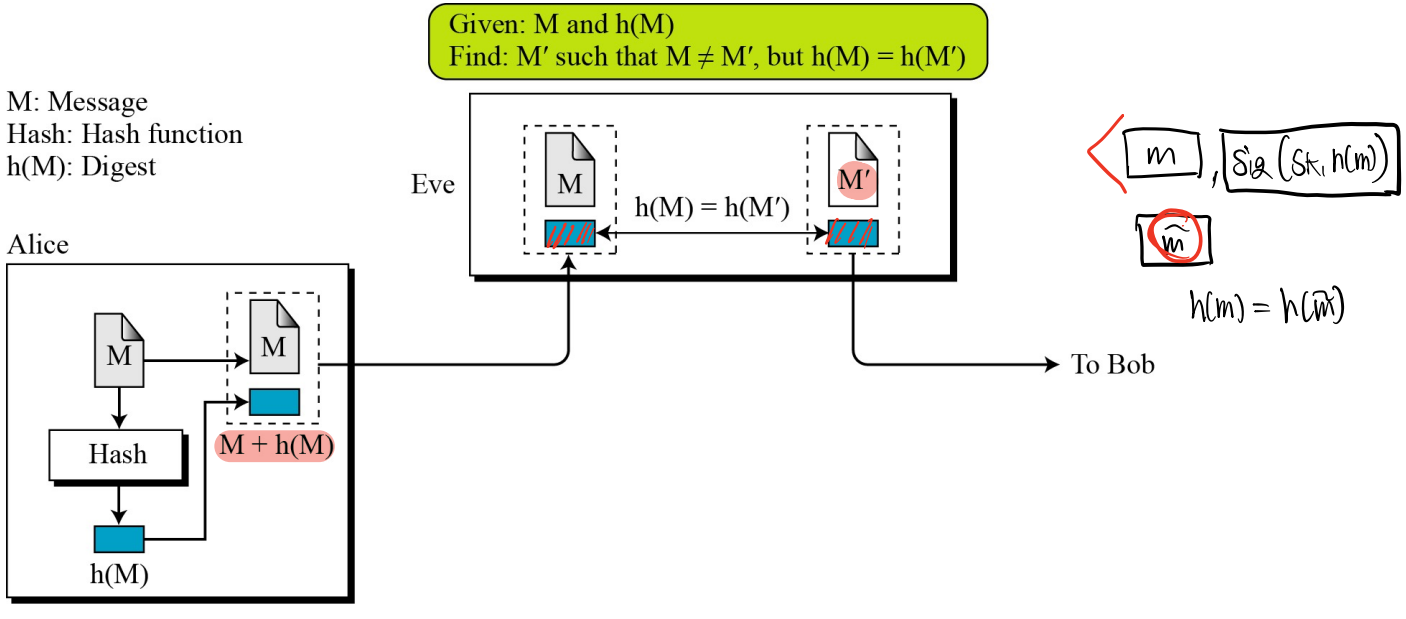


  - A CHF is *pre-image resistant* if it is infeasible for PPT adversary to find a message M' such that $y = h(M')$

Requirements for CHF (2)



- Second pre-image resistance
  - In real scenario (e.g., digital signature, MAC, public key encryption)



  - A CHF is *second pre-image resistant* if it is infeasible for PPT adversary to find a message $M' \neq M$ such that $h(M') = h(M)$

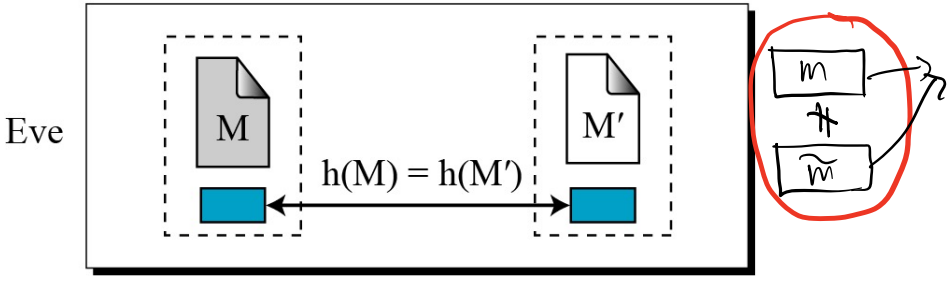
Requirements for CHF (3)

given $h(\cdot)$, find $x \neq \tilde{x}$ such that $h(x) = h(\tilde{x})$

- Collision resistance
  - In real scenario (e.g., digital signature, MAC)

M: Message
Hash: Hash function
h(M): Digest

Find: M and M′ such that M ≠ M′, but h(M) = h(M′)

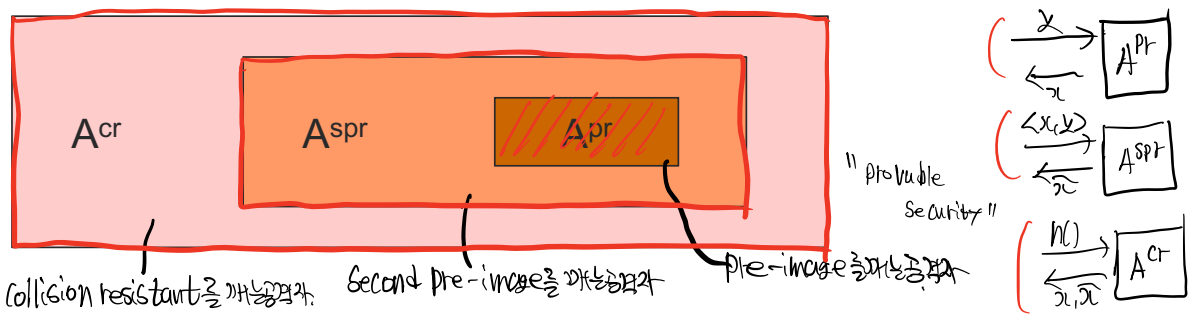


  - A CHF is *collision resistant* if it is infeasible for PPT adversary to find two distinct messages $M' \neq M$ such that $h(M') = h(M)$
  - By this property, the output of CHF can be used as 'fingerprint' or 'representative value of large data'

Relation between three requirements

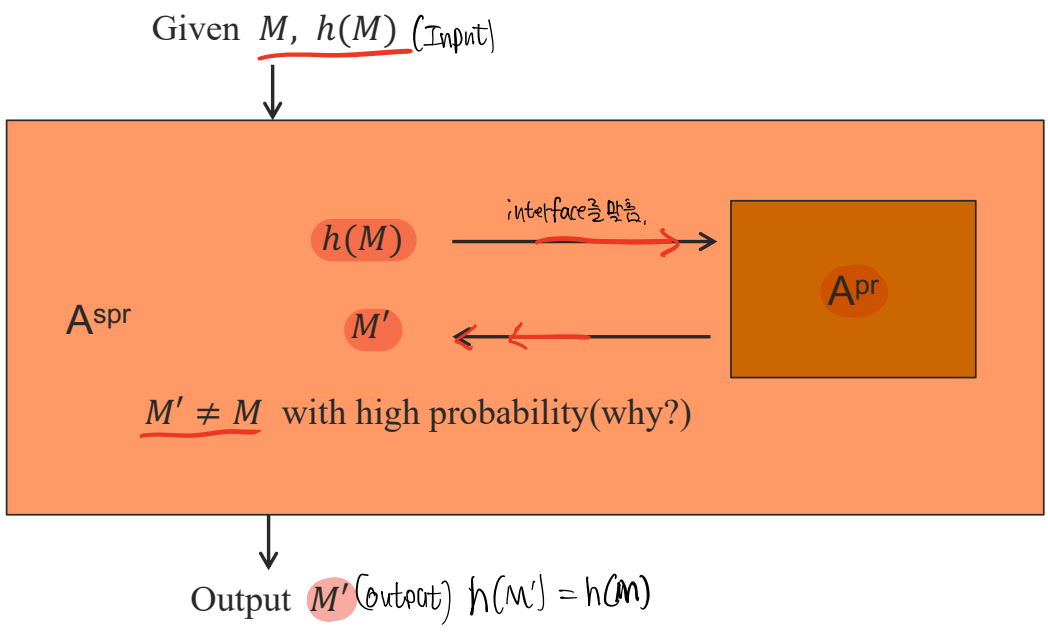
- Three requirements form a hierarchy
  - Pre-image resistant ← Second pre-image resistant ← Collision resistant
- What does the relation mean?
  - CHF is pre-image resistant ← CHF is second pre-image resistant ← CHF is collision resistant
  - Meaning 'collision resistance is the strong requirement to achieve'
  - Almost all of CHFs aim at achieving collision resistance

- Using PPT adversaries (A^{pr}, A^{spr}, A^{cr}), how can we prove that?



Proof of relation I

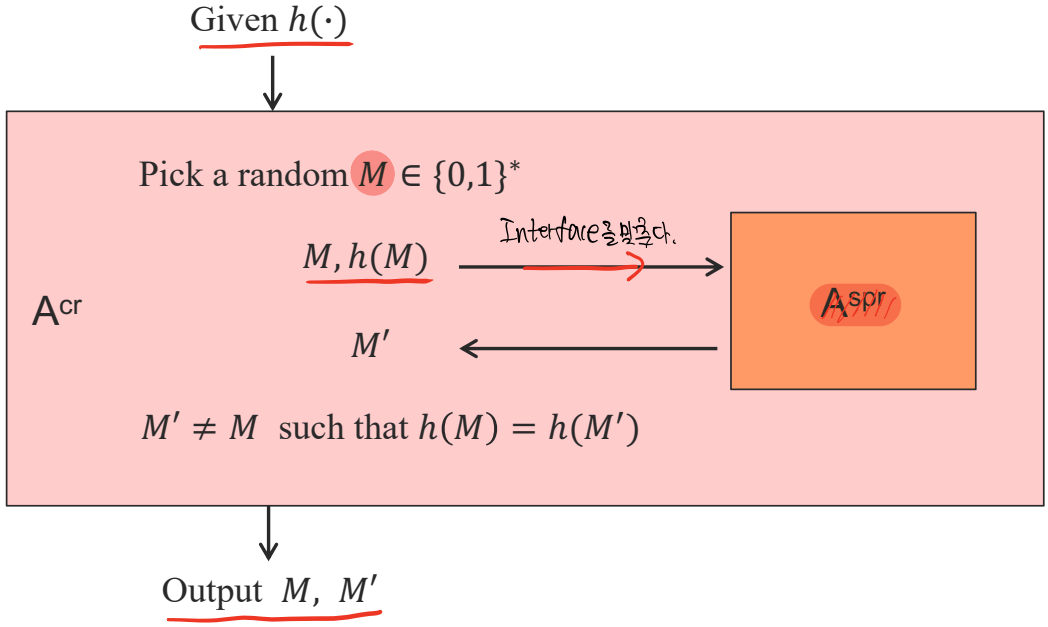
- CHF is pre-image resistant ← CHF is second pre-image resistant
  - Equivalence: If there is A^{pr}, then we can build A^{spr}
  - Proof:



  - A^{spr} can output M' such that $h(M) = h(M')$

Proof of relation II

- CHF is second pre-image resistant ← CHF is collision resistant
  - Equivalence: If there is A^{spr}, then we can build A^{cr}
  - Proof:



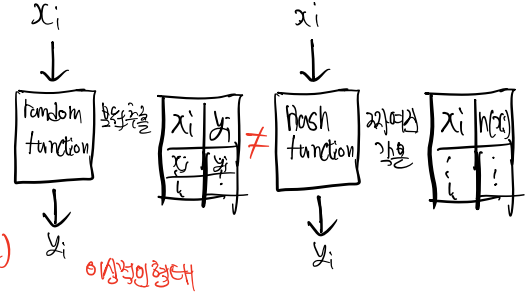
  - A^{cr} can find a collision (M, M') in $h(\cdot)$

A Generic 'Birthday' Attack

Hash 함수 설계시 충돌저항성만 보장된다.

알고리즘과 수학적으로 증명하는 증명.

- Probability analysis of finding a collision
  - Not relating to internal structures of any CHF
  - Analyzing mostly collision resistance (CR)
  - Giving a minimal bound of output length (아웃풋의 길이)
  - Considering CHF as a (idealized) random function



이상적인 형태

hash 함수가 일관적하는 이상적인 형태.

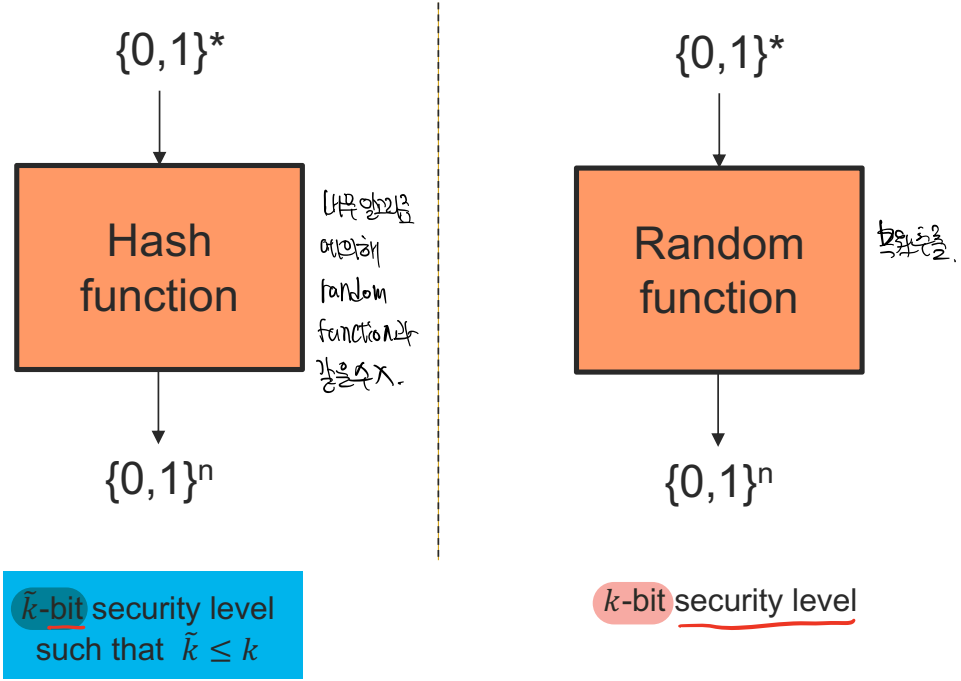
    - Random function $y = h(x)$
      - Given x as input, $h(x)$ is determined by selecting a value that is uniformly and independently distributed in $\{0,1\}^n$
    - Why do we consider an idealized random function?

실제 해쉬함수를 가지고 알고리즘에 의해 작성된 그런 것으로 사용할 때 해쉬함수로는 충돌저항성을 수학적으로 보장X.

- Finding a collision (under a random function)
  - In case of a random function: to be *likely* that $y_i = y_j$, *how many* $\{y_i\}$ *values* should be required ?
    - Assume all $\{x_i\}$ values all distinct
    - 'Likely' refers to that Pr[collision] ≥ 1/2

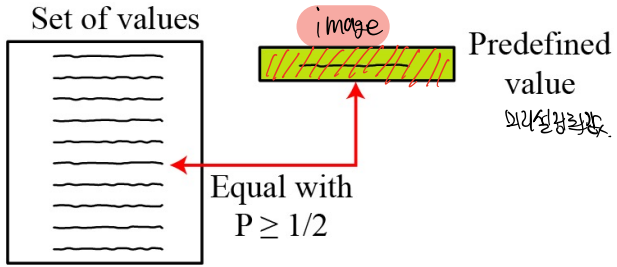
Why Random Function?

- Random function is a idealized model of any hash function
  - Hard to directly analyze the security of a hash function, based on analysis-driven approach
  - Instead, analyze a random function

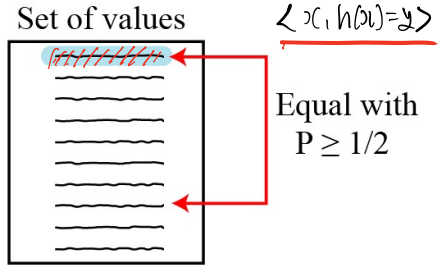


Birthday Problems

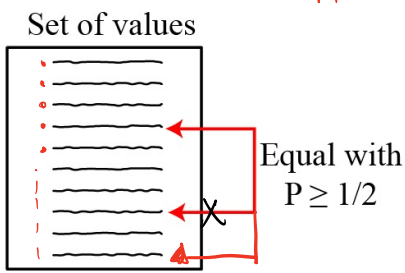
- Three ~~Four~~ birthday problems
  - What is the minimum number $k \in \{1, \ldots, 365\}$ of students in (a), (b), and (c) cases, respectively ?



a. First problem



b. Second problem



c. Third problem

Solutions of Birthday Problems (1)

- Each one is similar to:
  - Problem 1 ≈ attack against pre-image resistance
  - Problem 2 ≈ attack against second pre-image resistance
  - Problem 3 ≈ attack against collision resistance

Summarized solutions to four birthday problems

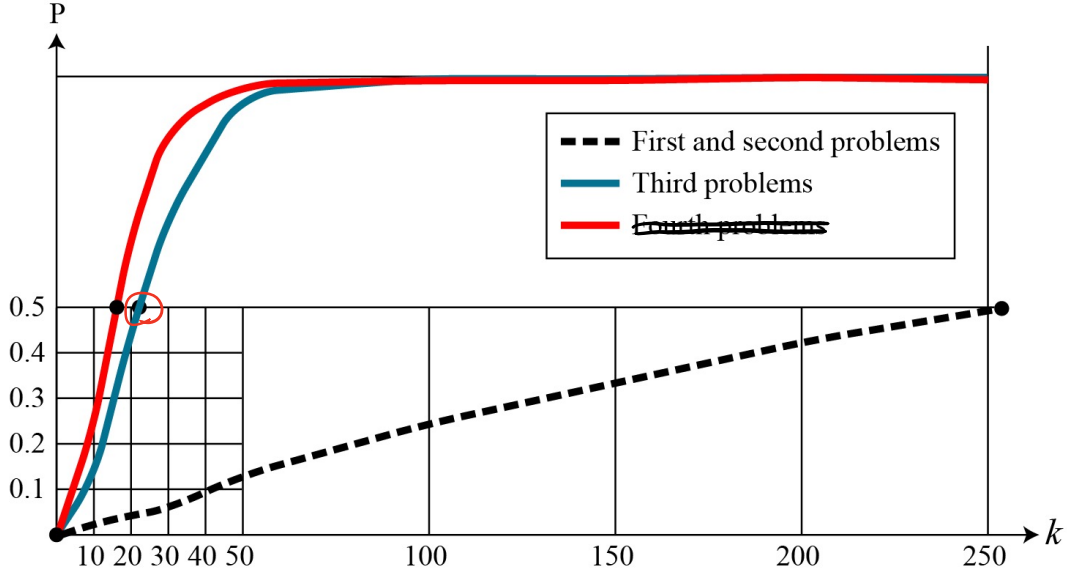
$N = 365$

Problem	*Probability*	*General value for k*	*Value of k with P = 1/2*	*Number of students (N = 365)*
1	$P \approx 1 - e^{-k/N}$	$k \approx \ln[1/(1-P)] \times N$	$k \approx 0.69 \times N$	253
2	$P \approx 1 - e^{-(k-1)/N}$	$k \approx \ln[1/(1-P)] \times N + 1$	$k \approx 0.69 \times N + 1$	254
3	$P \approx 1 - e^{k(k-1)/2N}$	$k \approx \{2 \ln [1/(1-P)]\}^{1/2} \times N^{1/2}$	$k \approx 1.18 \times N^{1/2}$	23

$1.18\sqrt{N}$

Solutions of Birthday Problems (2)

- Comparison between four birthday problems
  - N = 365
  - *'Likely'* means Pr[collision] ≥ ½
  - In 3rd birthday problem: Pr[collision] = $1 - \frac{365 \times 364 \times \cdots \times (365-k+1)}{365^k}$



Attacks on Random Function

- In attacking a random function $h: \{0,1\}^* \rightarrow \{0,1\}^n$ ($N = 2^n$)

Levels of difficulties for each type of attack

Attack	*Value of k with P=1/2*	*Order*
Preimage	$k \approx 0.69 \times 2^n$	2^n
Second preimage	$k \approx 0.69 \times 2^n + 1$	2^n
Collision	$k \approx 1.18 \times 2^{n/2}$	$2^{n/2}$

random function에서 충돌쌍을 찾기 위해서 hash함수를 시행해야하는 횟수.

  - Showing that attacking collision resistance is much easier than pre-image and second pre-image attacks
  - Example: a CHF $h(x)$ with output length of 64 bits
    - $k = 1.18 \times 2^{n/2} = 1.18 \times 2^{32}$. If attacker can do hash evaluations 2^{20} 평균적. inputs per second, it takes 1.18×2^{12} seconds (less than 2 hours)

$1.18 \times 2^{32} / 2^{20}$ seconds.

$= 1.18 \times 2^{12}$ s $\approx$ 2hour.

Example of Collision Attacks

- MD5 (with digest of 128 bits)
  - Which was one of the standard hash functions for a long time
  - To launch a collision attack, adversary needs to test 2^{64} ($2^{128/2}$) tests
  - Even if adversary can perform 2^{30} tests in a second, it takes 2^{34} seconds (more than 500 years) - based on the random function
  - It has been proved that MD5 can be attacked on much less than 2^{64} tests (in minutes) because of the structure of the algorithm random이 아닌 real hash함수에 대한 구조가 있다.
- SHA-1 (with digest of 160 bits)
  - Which was one of the standard hash functions developed by NIST
  - To launch a collision attack, adversary needs to test $2^{160/2} = 2^{80}$ tests
  - Even if adversary can perform 2^{30} tests in a second, it takes 2^{50} seconds (more than ten thousand years) – based on the random function
  - In 2005, Chinese researchers have discovered that SHA-1 requires time 2^{69} (still theoretically) , which is in less time than 2^{80} $2^{69} < 2^{80}$ Break
    - Currently, move towards SHA-2 family (SHA-256, SHA-512) and new design SHA-3

Sample CRHFs

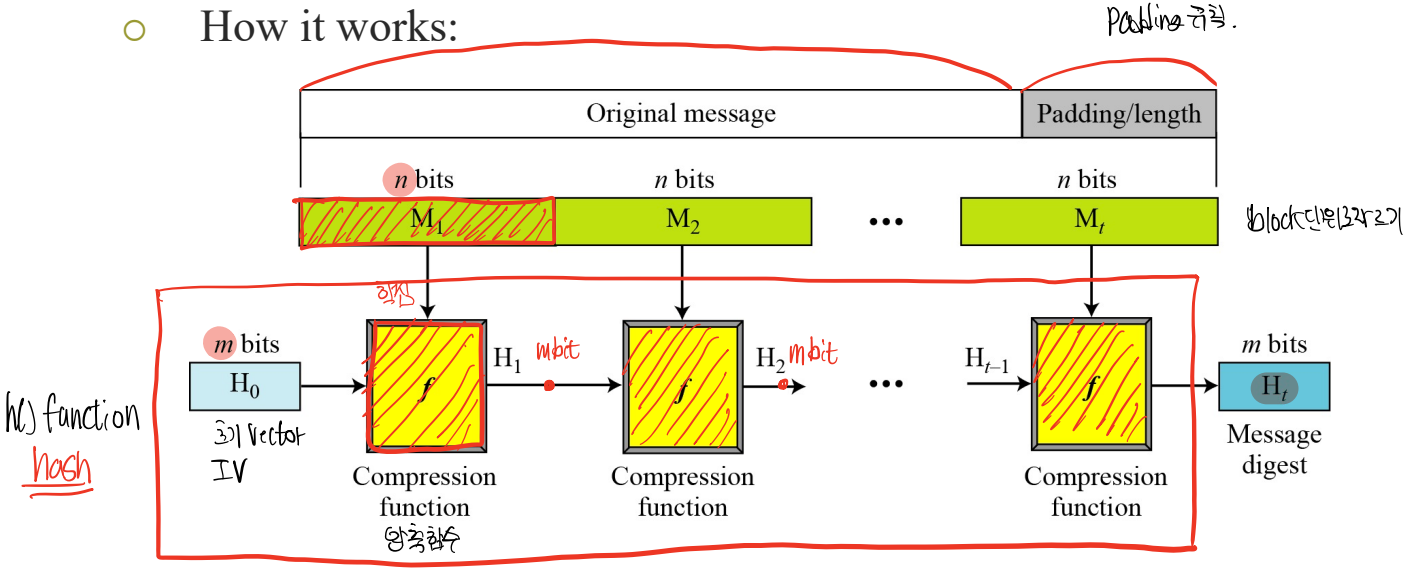
- AMD Opteron, 2.2 GHz (Linux)

	function	digest size (bits)	Speed (MB/sec)	generic attack time
NIST standards	SHA-1	160	153	2^{80}
	SHA-256	256	111	2^{128}
	SHA-512	512	99	2^{256}
	Whirlpool	512	57	2^{256}

* best known collision finder for SHA-1 requires 2^{51} hash evaluations

How to Construct CRHF

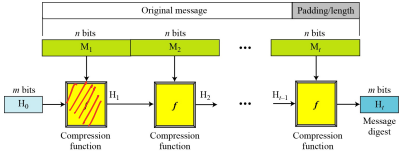
- The Merkle-Damgard transform
  - How it works:



  - Theorem: If $f(\cdot)$ is collision-resistant, then $H(\cdot)$ is collision-resistant
    - Can restrict our attention to the fixed-length $f(\cdot)$
    - $f(\cdot)$ is often called 'compression function'
    - H_0 (a.k.a. IV) is usually fixed

Two Groups of Compression Functions (1)

- CRHF based on specially designed compression functions
  - MD2, MD4, and MD5: all broken
  - SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512
    - All of SHA series have similar structure
  - RIPEMD-160, HAVAL, HAS(Korean Standard), …

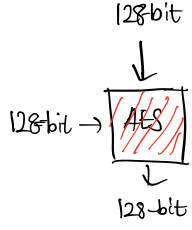


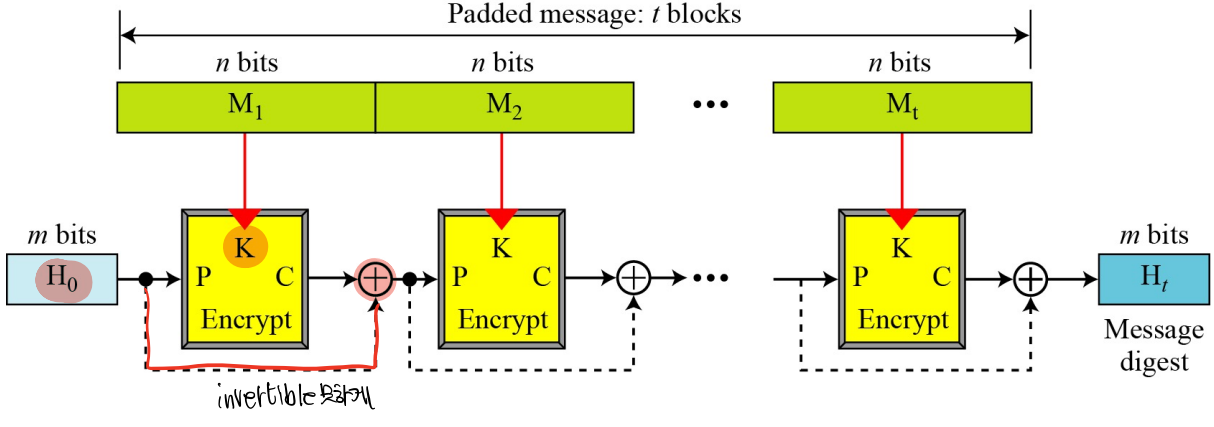
Characteristics of Secure Hash Algorithms (SHAs)

Characteristics	*SHA-1*	*SHA-224*	*SHA-256*	*SHA-384*	*SHA-512*
Maximum Message size	$2^{64} - 1$	$2^{64} - 1$	$2^{64} - 1$	$2^{128} - 1$	$2^{128} - 1$
Block size	512	512	512	1024	1024
Message digest size	160	224	256	384	512
Number of rounds	80	64	64	80	80
Word size	32	32	32	64	64

Two Groups of Compression Functions (2)

- CRHF based on block ciphers
  - Instead of creating a new compression function, employ AES
  - Only encryption algorithm (in block cipher) is used
  - Whirlpool, Davies-Meyer, Matyas-Meyer-Oseas, …
  - Example: Davies-Meyer scheme

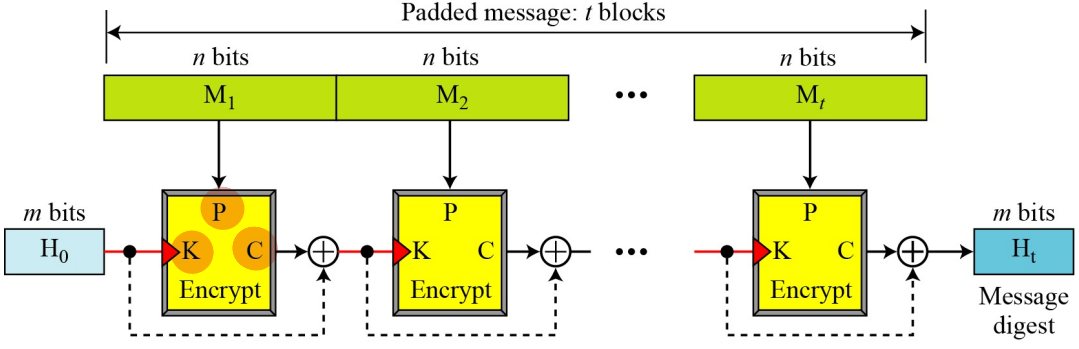




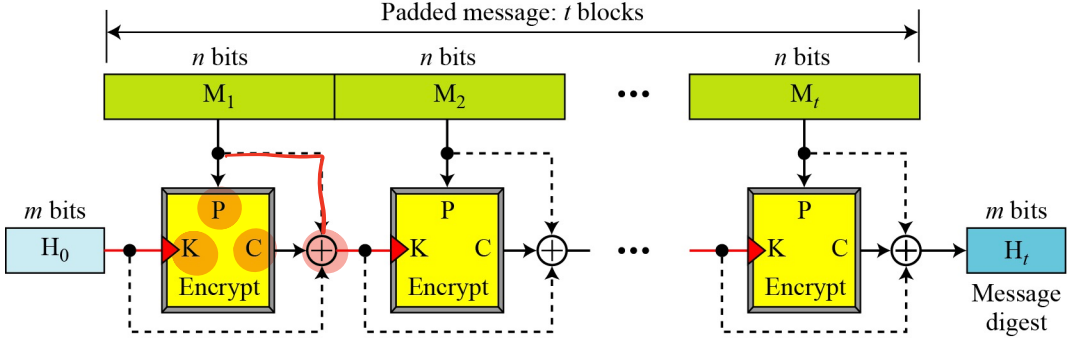
  - What is the main advantage of CRHF based on block ciphers?
  - What if Xor operation does not exist? (assuming H_0 is part of messages)

More CRHFs based on Block Ciphers (1)

- Matyas-Meyer-Oseas scheme
  - Used if |K|=|C|, like AES (128-bit version)

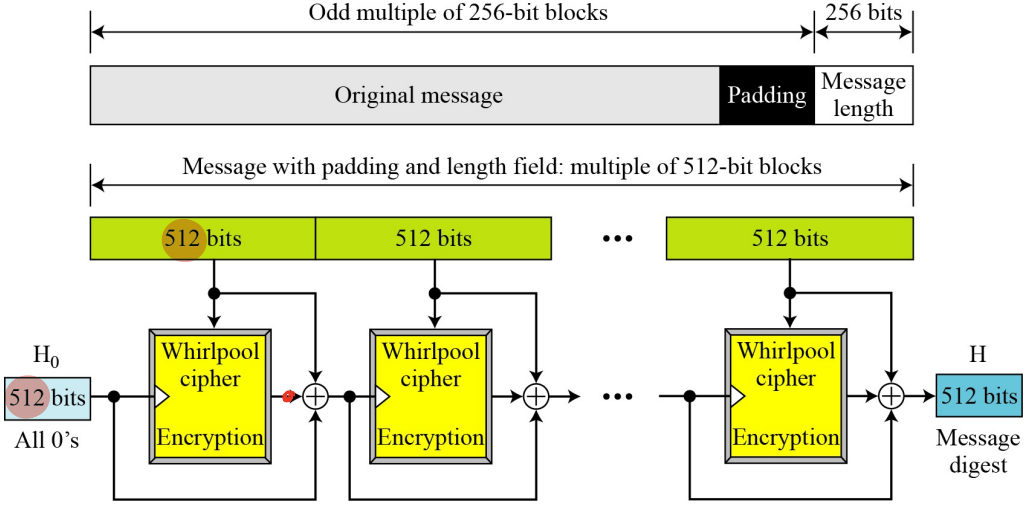


- Miyaguchi-Preneel scheme



More CRHFs based on Block Ciphers (2)

- Overview of Whirlpool

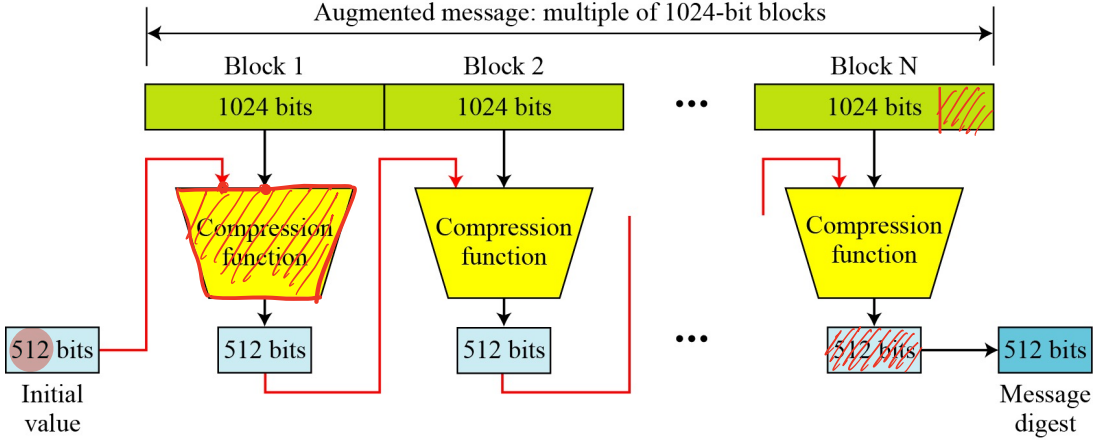


  - Based on Miyaguchi-Preneel scheme
  - Block cipher is a modified AES that is tailored for CRHF
  - |K|=|P|=|C|=512 bits
  - Endorsed by European researchers

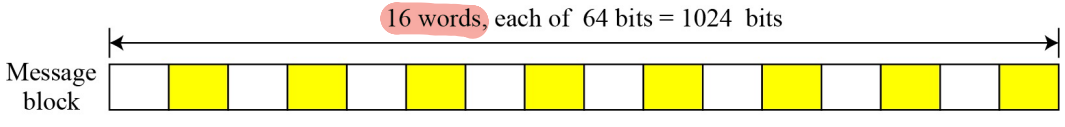
Overview of SHA-512

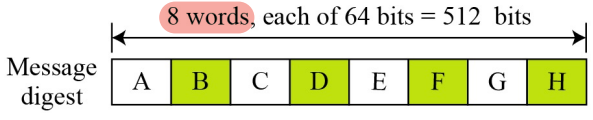
https://github.com/intel/tinycrypt

- Based on the Merkle-Damgard transform
  - IV(512 bits) is predetermined



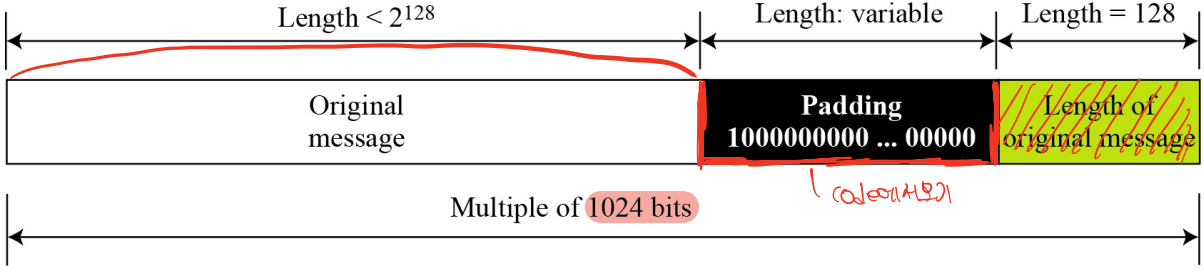
  - Word-oriented: (word = 64 bits) 256 = 32bit .





SHA-512 (1)

- Length field and padding
  - Assign an integer of length between 0 and 2^{128}-1



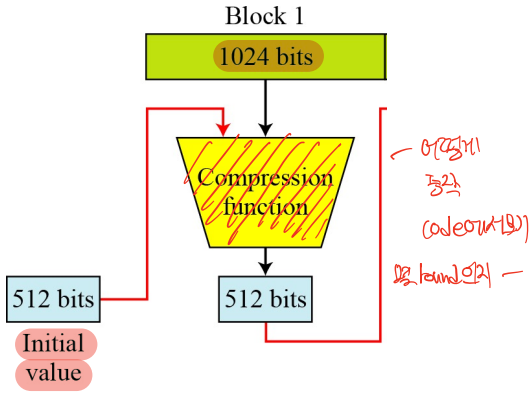
  - What if the length of message is bigger than 2^{128} bits?

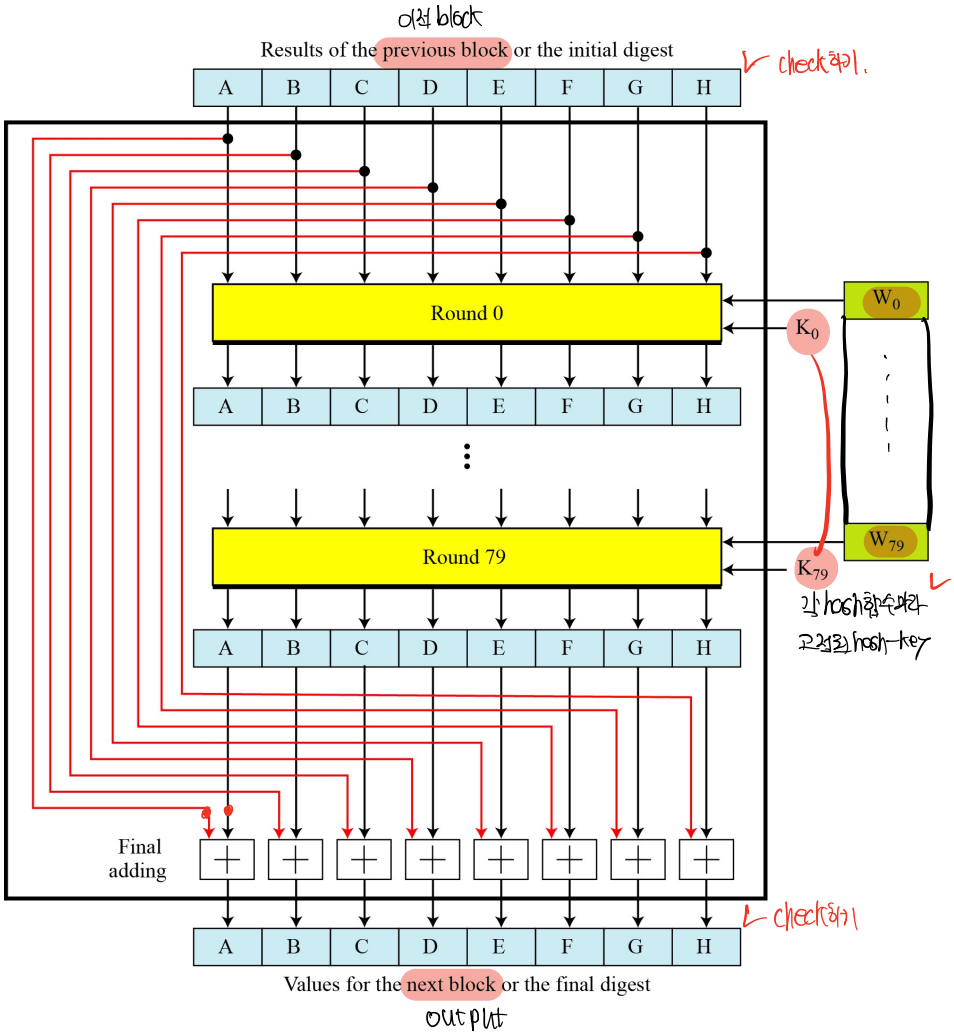
- Initialization value: 8-words = 512 bits

Buffer	*Value (in hexadecimal)*	*Buffer*	*Value (in hexadecimal)*
A_0	**6A09E667F3BCC908**	E_0	**510E527FADE682D1**
B_0	**BB67AE8584CAA73B**	F_0	**9B05688C2B3E6C1F**
C_0	**3C6EF372EF94F828**	G_0	**1F83D9ABFB41BD6B**
D_0	**A54FE53A5F1D36F1**	H_0	**5BE0CD19137E2179**

SHA-512 (2)

- Compression function

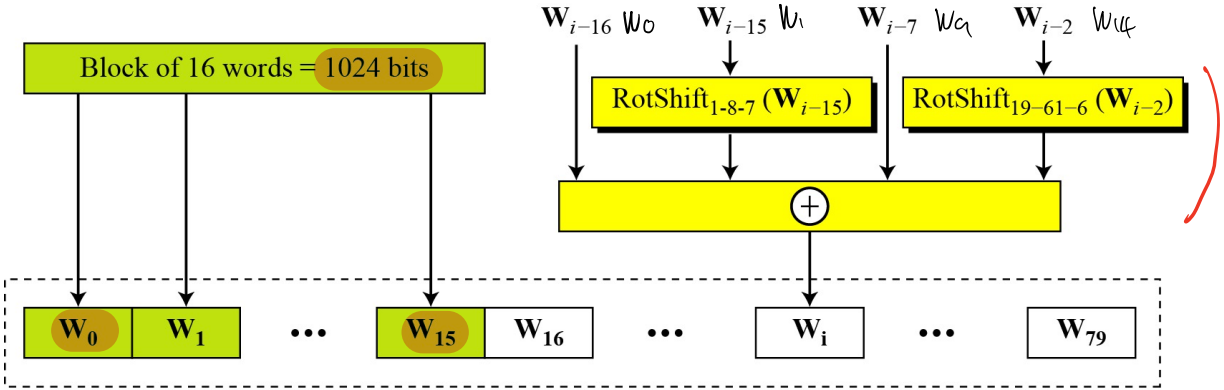




- 1024 bits (16 words)
  are expanded to 80 words
- ⊞ addition (mod 2^{64})

SHA-512 (3)

- Word expansion (16 words → 80 words)



RotShift$_{l\text{-}m\text{-}n}$ (x): RotR$_l$ (x) ⊕ RotR$_m$ (x) ⊕ ShL$_n$ (x)

RotR$_i$ (x): Right-rotation of the argument x by i bits
ShL$_i$ (x): Shift-left of the argument x by i bits and padding the left by 0's.

- Each word from W_{16} to W_{79} is made from 4 previous words.
- For instance, W_{60} is made as:

$$W_{60} = W_{44} \oplus \text{RotShift}_{1\text{-}8\text{-}7}(W_{45}) \oplus W_{53} \oplus \text{RotShift}_{19\text{-}61\text{-}6}(W_{58})$$

SHA-512 (4)

- $\text{RotShift}_{1\text{-}8\text{-}7}(W_i) = RotR_1(W_i) \oplus RotR_8(W_i) \oplus ShL_7(Wi)$
  - W_i = 1010001000010 ⋯ 01110111001 (64 bits)
  - $\text{RotR}_1(W_i)$ = 1101000100010 ⋯ 0111011100
  - $\text{RotR}_8(W_i)$ = 1011100110100010010 ⋯ 011
  - $\text{ShL}_7(Wi)$ = 00010 ⋯ 0111011100100000000
- $\text{RotShift}_{19\text{-}61\text{-}6}(W_i) = RotR_{19}(W_i) \oplus RotR_{61}(W_i) \oplus ShL_6(Wi)$ is performed in a similar way

SHA-512 (5)

- 80 constants (K_0 to K_{79}) used for 80 rounds (each one = 64 bits)

Eighty constants used for eighty rounds in SHA-512

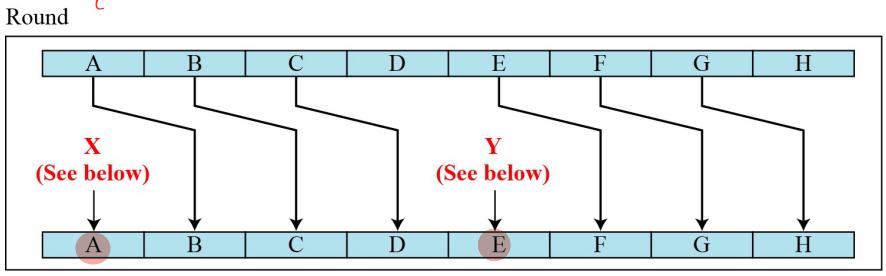
428A2F98D728AE22	7137449123EF65CD	B5C0FBCFEC4D3B2F	E9B5DBA58189DBBC
3956C25BF348B538	59F111F1B605D019	923F82A4AF194F9B	AB1C5ED5DA6D8118
D807AA98A3030242	12835B0145706FBE	243185BE4EE4B28C	550C7DC3D5FFB4E2
72BE5D74F27B896F	80DEB1FE3B1696B1	9BDC06A725C71235	C19BF174CF692694
E49B69C19EF14AD2	EFBE4786384F25E3	0FC19DC68B8CD5B5	240CA1CC77AC9C65
2DE92C6F592B0275	4A7484AA6EA6E483	5CB0A9DCBD41FBD4	76F988DA831153B5
983E5152EE66DFAB	A831C66D2DB43210	B00327C898FB213F	BF597FC7BEEF0EE4
C6E00BF33DA88FC2	D5A79147930AA725	06CA6351E003826F	142929670A0E6E70
27B70A8546D22FFC	2E1B21385C26C926	4D2C6DFC5AC42AED	53380D139D95B3DF
650A73548BAF63DE	766A0ABB3C77B2A8	81C2C92E47EDAEE6	92722C851482353B
A2BFE8A14CF10364	A81A664BBC423001	C24B8B70D0F89791	C76C51A30654BE30
D192E819D6EF5218	D69906245565A910	F40E35855771202A	106AA07032BBD1B8
19A4C116B8D2D0C8	1E376C085141AB53	2748774CDF8EEB99	34B0BCB5E19B48A8
391C0CB3C5C95A63	4ED8AA4AE3418ACB	5B9CCA4F7763E373	682E6FF3D6B2B8A3
748F82EE5DEFB2FC	78A5636F43172F60	84C87814A1F0AB72	8CC702081A6439EC
90BEFFFA23631E28	A4506CEBDE82BDE9	BEF9A3F7B2C67915	C67178F2E372532B
CA273ECEEA26619C	D186B8C721C0C207	EADA7DD6CDE0EB1E	F57D4F7FEE6ED178
06F067AA72176FBA	0A637DC5A2C898A6	113F9804BEF90DAE	1B710B35131C471B
28DB77F523047D84	32CAAB7B40C72493	3C9EBE0A15C9BEBC	431D67C49C100D4C
4CC5D4BECB3E42B6	4597F299CFC657E2	5FCB6FAB3AD6FAEC	6C44198C4A475817

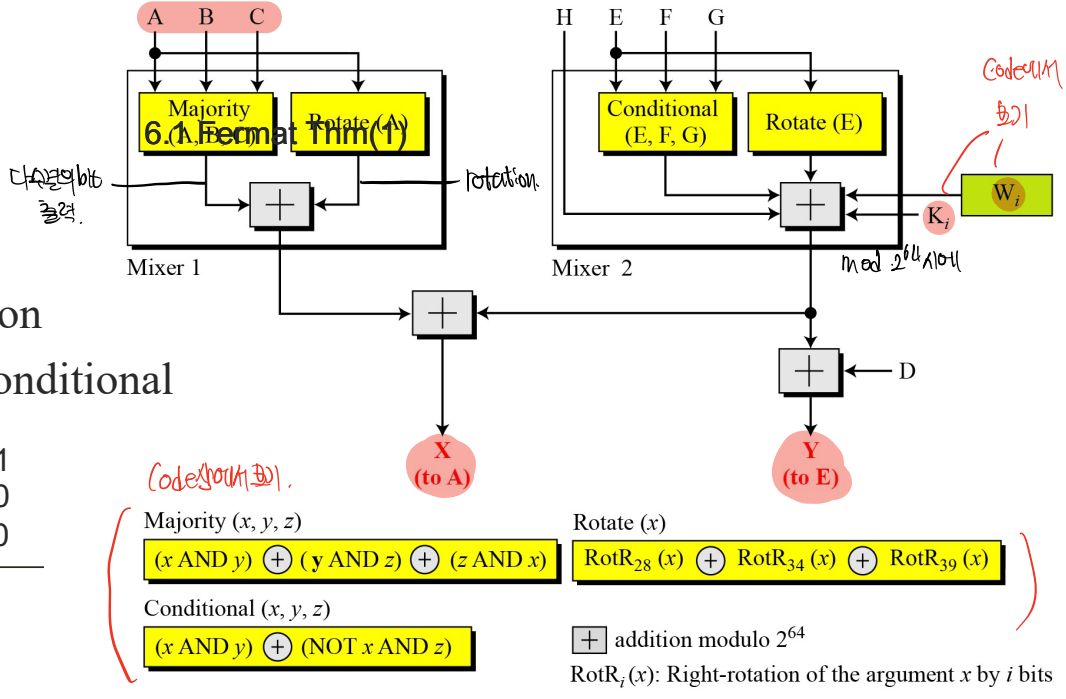
SHA-512 (6)

- Round function
  - AND, NOT
    - 1 AND 1 = 1
    - 0 AND 1 = 0
    - NOT 1 = 0
  - Bit by bit operation

    in Majority and Conditional

A = 100110 ··· 001001
B = 011001 ··· 110100
C = 101110 ··· 011010



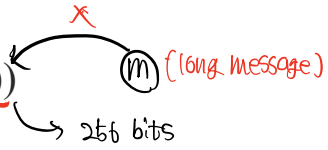


Homework 2

- https://github.com/intel/tinycrypt
  - SHA-256 code analysis (due date: 10/18)

Uses of CHF

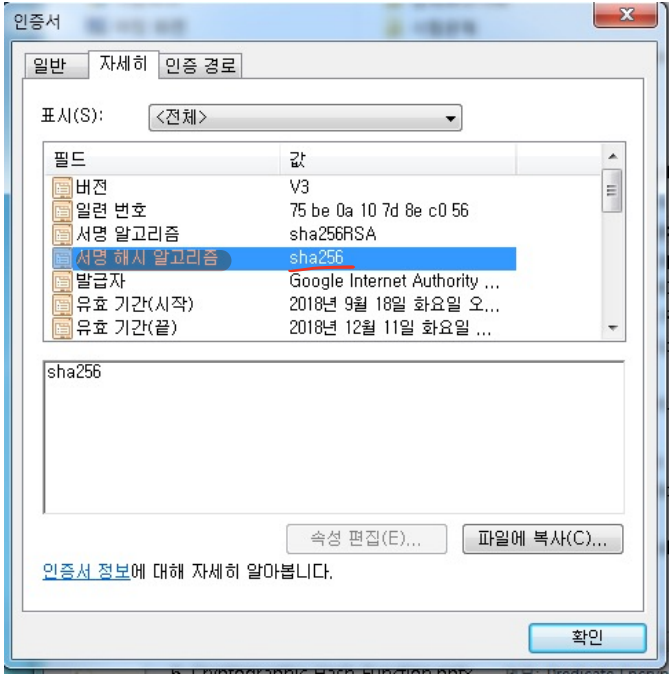
- CHFs are widely used in cryptography
  - Digital signatures
    - Hash-then-sign: Sign(SK, H(m))

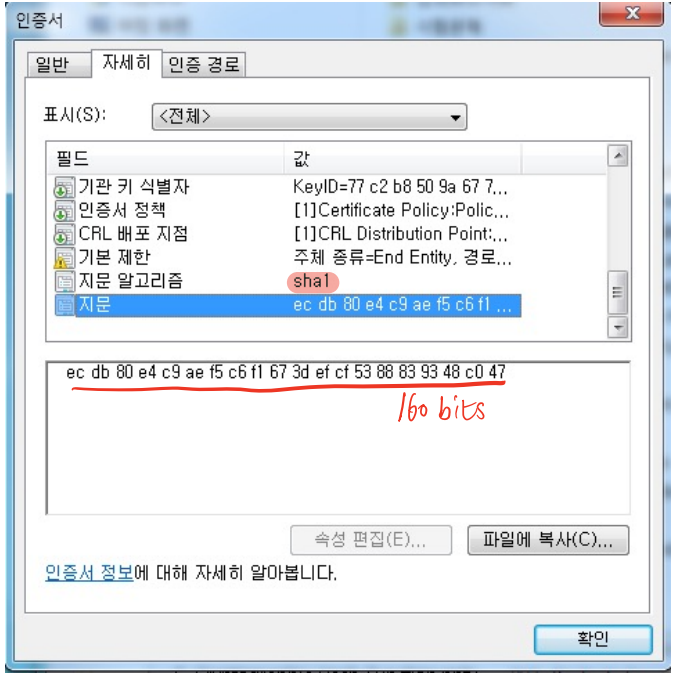


  - Public key encryption systems
    - RSA-PKCS#1 v1.5
    - RSA-OAEP (Optimal Asymmetric Encryption Padding)
  - Message authentication code (MAC)
    - HMAC
  - Applications where pseudo-randomness is required
    - PRG (Pseudo-Random Generator)
    - KDF (Key Derivation Function) – HKDF / PBKDF
  - Blockchain
    - …

Example of Hash-then-Sign

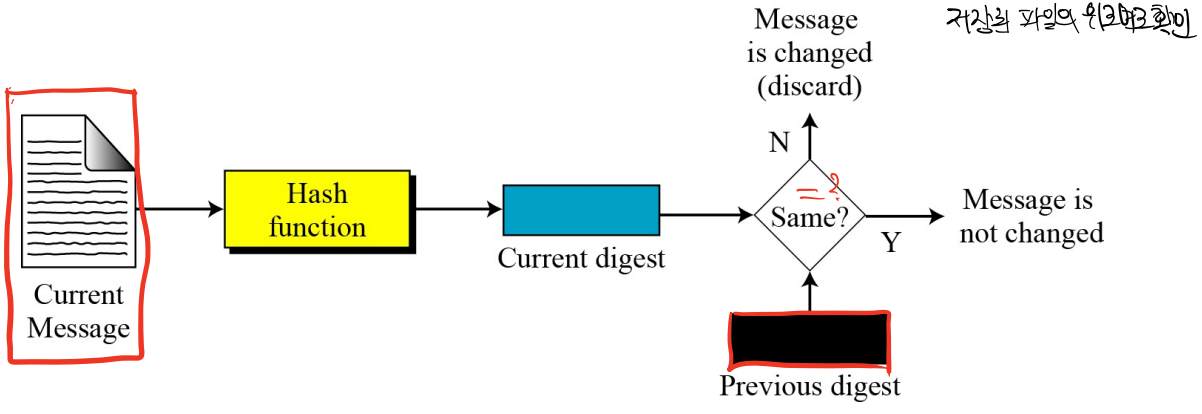
- In a certificate for authenticity





Integrity of stored data(1)

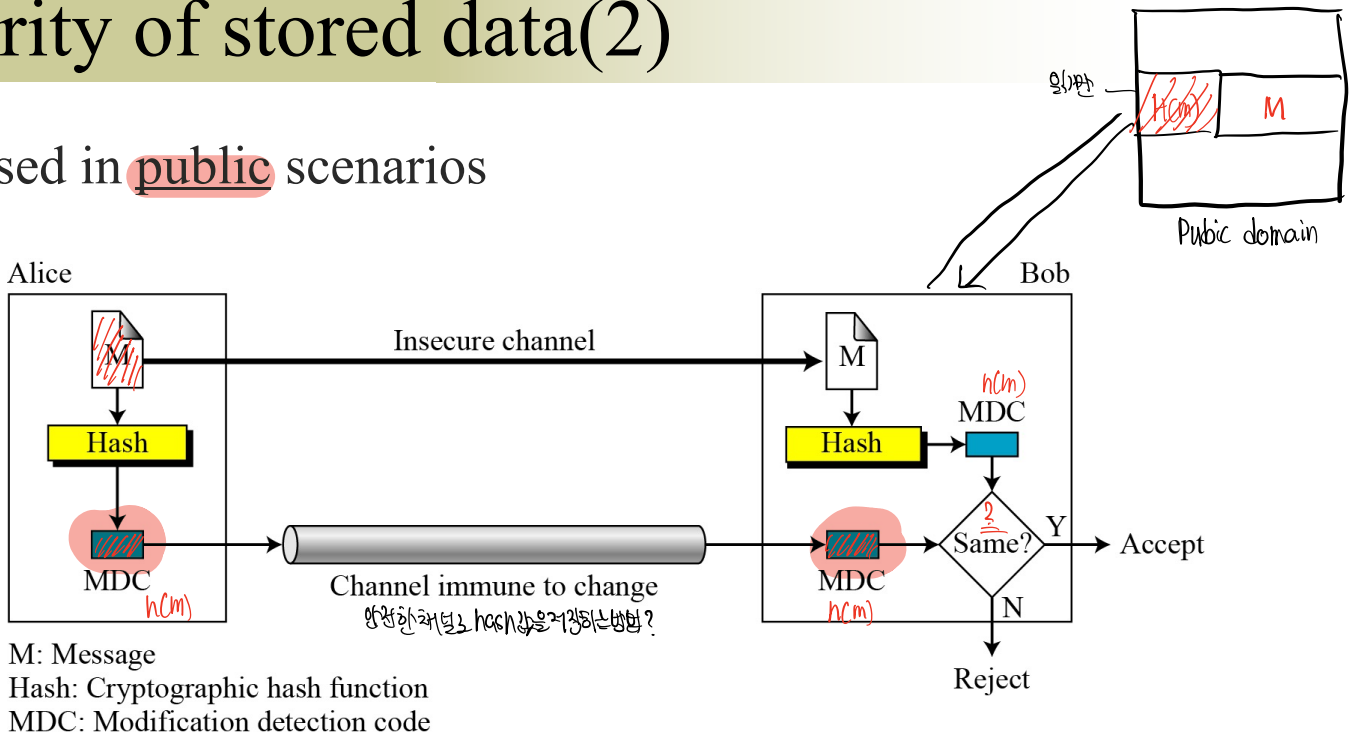
- Used in non-transmission & non-public scenarios
  - Checking integrity of stored data
  - Checking integrity of data encryption key stored



  - Changing even a single bit will be noticed by the property of CHF

Integrity of stored data(2)

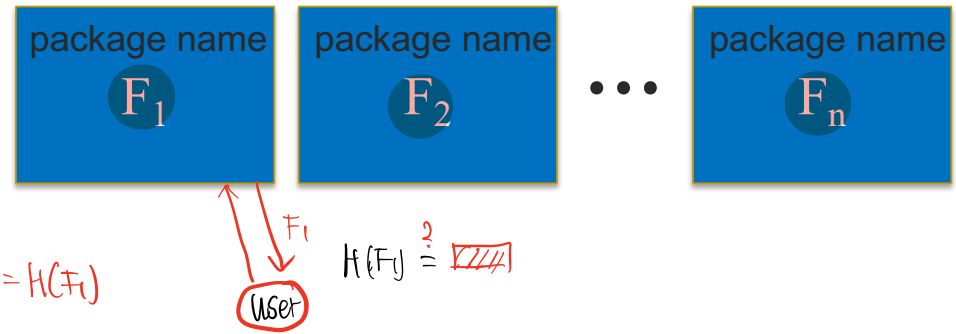
- Used in public scenarios

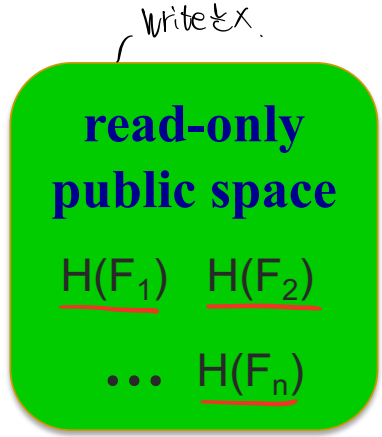


- H(M) should be sent (or stored) in a safe channel
- Changing even a single bit will be noticed by the property of CHF
- Application scenario:
  - Checking integrity of distributed software, notarization, seized data, …

Integrity of files

- Protecting file integrity using CRHF H
  - Software packages:





  - When user downloads package, can verify that contents are valid
  - H is collision resistant ⇒

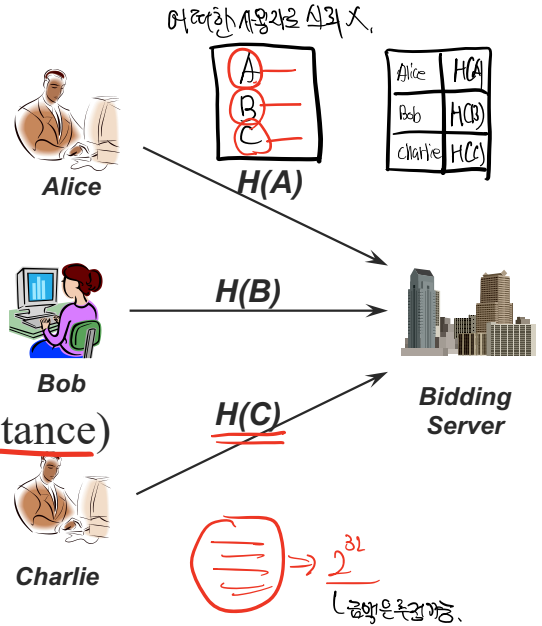
    attacker cannot modify package without detection

  - No key is needed (public verifiability), but requires read-only space
  - Linux

Online bidding (입찰)

- Online bids using CRHF H
  - Alice, Bob, and Charlie want to place bids
  - They don't trust each other and even more the bidding server

- A possible solution ?
  - Alice, Bob, Charlie hashes H(A), H(B), H(C)
  - All hashes are submitted and posted online
  - Then bids A, B, C are revealed later
  - Hash doesn't reveal bids (one-way)
  - Can't change bid after hash sent (collision-resistance)
  - But there is a flaw here…



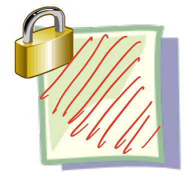
- How to fix the flaw?

숙제 Homework 3. , ppt 1장으로 간단히 설명.
10.10일까지

KDF(Key Derivation Function)

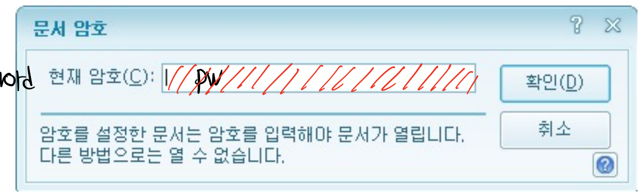
- Alice wants to encrypt a file
  - Using a symmetric-key cipher like 3DES or AES
  - How can we have access to a key K_A ?

CBC/CTR



K_A

- Consider two methods:
  - The key is generated as $K_A = h(pw)$
  - Actually, $h(pw) \rightarrow$ K_A and IV
  - Zip-/ pdf-/ hwp-encryption

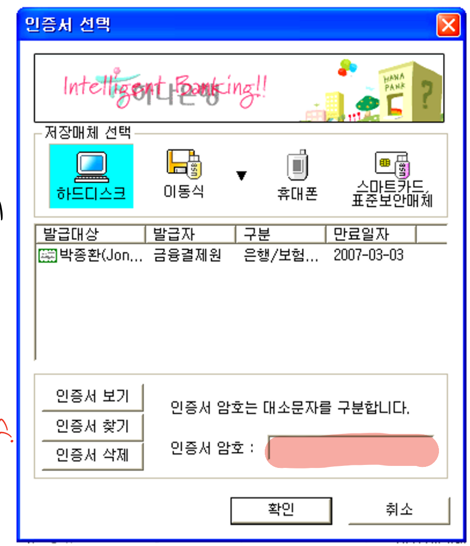


  - K is randomly chosen, and then K is stored as

$CT = E(K_A, K)$, where $K_A = h(pw)$

C=Enc(K, m)

  - Certificate-based encryption (pdf-/ hwp-)



- What are differences between them?
  - In terms of storage and security level

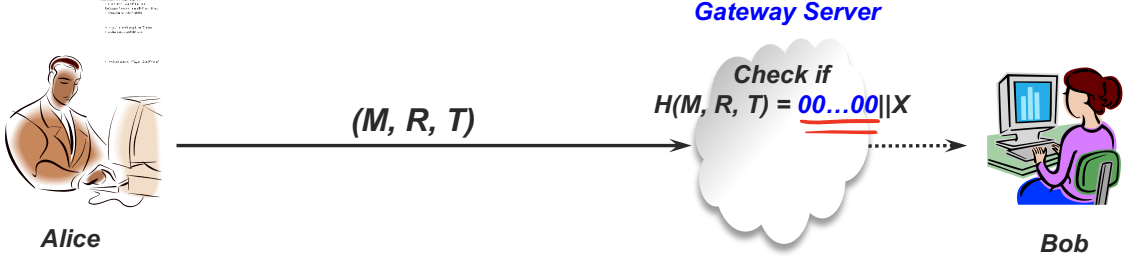
Spam Reduction

- Spam reduction technique using CHF
  - Alice sends an email message M at the time T
  - M = (message, Alice's email address, Bob's email address)
  - Alice must determine a value R such that

$$H(M, R, T) = (00\ldots00 \,||\, X)$$

($00\ldots00$: N bits)

  - On average, Alice needs to compute about 2^N hashes
    - N is determined by acceptable level of Bob



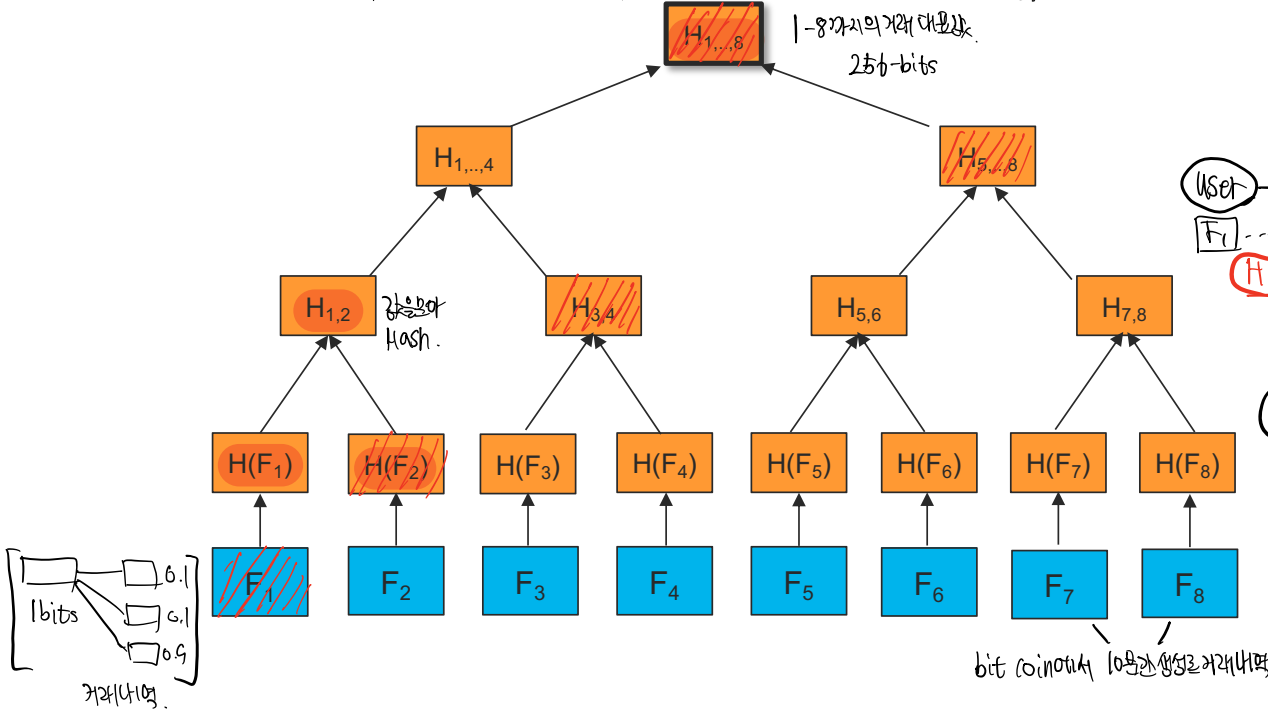
  - How about a spammer's work? (e.g., $N = 4$ and 2^{10} receivers)

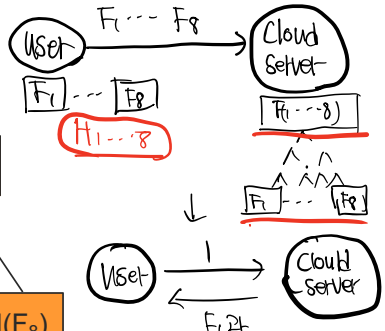
Merkle Trees

H: SHA-256

- Consider a client who uploads files {Fi} to a server
  - In case where the storage of client is small
    - Using Merkle trees, client can store only the root node value

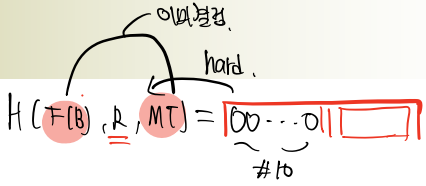
중요한점 : 1. root node를 갖고 있으면 leaf node가 hash함수의 collision에도 재귀적보호인가?
2. 유저의 위조변조여부를 확인하려면 해당유저와 log n 정도만 추가하면 check가능.

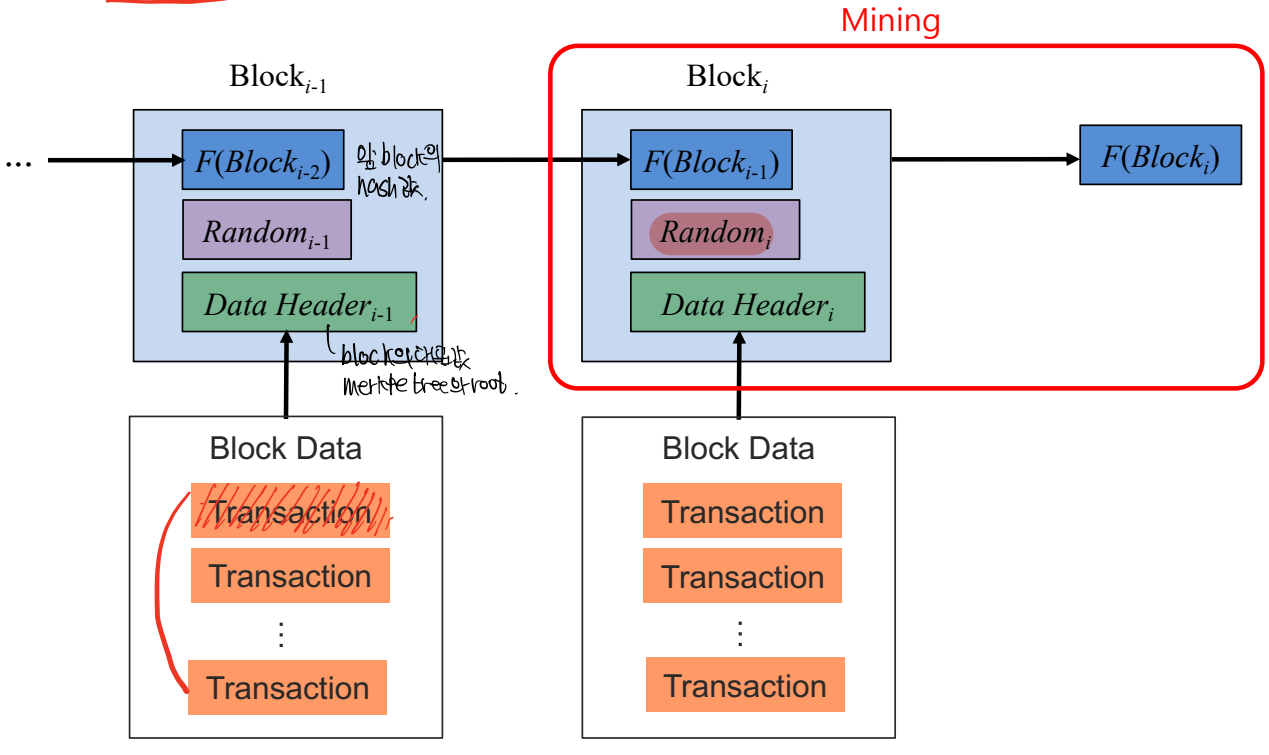




Blockchain

- PoW(Proof of Work)





- Mining example: $F(Block_i)$ = 00 … 0002fab…289d (as a hash output)

Q & A